CLAIMS

What is claimed is:

1. A policy compiler comprising:

a system controller to generate a configuration data abstraction layer of a routing policy, the configuration data abstraction layer to map configuration to an intermediate layer comprising fields, operators and arguments; and

a policy repository to verify the intermediate layer against a set of verification rules for one or more client protocols including versions thereof, the policy repository to generate compiled policy transmission language for use by the one or more client protocols including versions thereof.

2. The compiler of claim 1 wherein the policy repository is to verify the intermediate layer against the set of verification rules for a particular one or more of a plurality of client protocols including versions of the one or more of the plurality of client protocols, and

wherein the policy repository is to generate the compiled policy transmission language for use by the one or more client protocols including versions thereof.

3. The compiler of claim 2 wherein, for each statement in the policy, the policy repository is to verify fields for each of the client protocols including versions thereof, is to verify field-operator parings in the policy, and is to verify one or more arguments used for each field-operator pairing in the policy,

wherein the verifying is based on verification rules associated with a client dynamic link library (DLL) for each of the client protocols including versions thereof.

4. The compiler of claim 3 wherein the policy repository is to validate the policy against at least two versions of a border gateway protocol (BGP), and

wherein the compiled policy transmission language is to be executed by the at least two versions of the border gateway protocol.

5. The compiler of claim 3 wherein the policy repository is to further validate the policy against one or more versions of a border gateway protocol (BGP) client protocol, one or more versions of an open-shortest-path-first (OSPF) client protocol and one or more versions of an intermediate system to intermediate system (IS-IS) client protocol, and

wherein the compiler further comprises an execution engine associated with each of the one or more versions of the client protocols, the associated execution engines to separately execute the compiled policy transmission language of the associated client protocol.

6. The compiler of claim 3 wherein the policy repository is to perform a verification in response to a request for use of the policy for an attach point, the attach point being a set of capabilities associated with a version of one of the client protocols.

7. The compiler of claim 1 wherein the compiled policy transmission language comprises a set of rules for each policy statement of a verified policy for execution by an execution engine associated with ones of the client protocols and versions thereof when verified against ones of the client protocols and versions thereof.

8. The compiler of claim 1 further comprising an execution engine for each of a plurality of client protocols and versions thereof,

wherein the execution engines are to execute the compiled policy transmission language based on execution rules provided by an associated client dynamic link library (DLL).

9. The compiler of claim 8 wherein the execution rules provided by one of the client DLLs are used by an execution engine of the associated version of one of the client protocols during execution of the compiled policy.

10. The compiler of claim 8 wherein the system controller is to perform an optimization procedure to improve execution performance of the compiled policy transmission language,

wherein as part of the optimization procedure comprises rearranging policy statements to improve the execution performance,

wherein for a field-operator pair in an associated one of the client DLLs, the system controller is to determine when a statement is processing intensive, and is to rearrange at least some of the statements to check at least some processing intensive during execution after less processing intensive statements.

11. The compiler of claim 10 wherein as part of the optimization procedure, the system controller is to further eliminate at least some repeated statements or repeated portions of statements, and

wherein the policy repository is to generate an optimized compiled policy in policy transmission language prior to compiling.

12. The compiler of claim 1 wherein at least one of the versions of the client protocols is to mark route attributes in a batch of routes as invariant for route attributes that share values across differing route prefixes, and

wherein an execution engine associated with one of the client protocols is to cache results for statements associated with the marked attributes when executing the compiled policy transmission language,

wherein upon continued execution of the compiled policy transmission language, the execution engine is to use the cached results for evaluations of subsequent statements in the policy which reference the marked attributes.

13. A method of generating compiled policy for execution by one or more client protocols including one or more versions of the client protocols, the method comprising:

generating a configuration data abstraction layer for a routing policy to map configuration to an intermediate layer;

verifying the intermediate layer against a set of rules for the one or more client protocols including the one or more versions thereof; and

generating compiled policy transmission language when statements of the intermediate layer are verified against the set of rules for the one or more client protocols including the one or more versions thereof.

14. The method of claim 13 further comprising executing the compiled policy transmission language with an execution engine associated with the one or more client protocols including the one or more versions thereof, the executing being based on execution rules for the one or more client protocols including the one or more versions thereof, the execution rules being provided by a client dynamic link library (DLL) associated with the one or more client protocols including the one or more versions thereof.

15. The method of claim 14 wherein validating comprises verifying the routing policy against a border gateway protocol (BGP) client protocol, an open-shortest-path-first (OSPF) client protocol and an intermediate system to intermediate system (IS-IS) client protocol, and

wherein executing further comprises separately executing, with an execution engine associated with each of the client protocols including versions thereof, the compiled policy transmission language.

16. The method of claim 13 wherein generating compiled policy transmission language comprises performing an optimization procedure to improved execution performance of the compiled policy transmission language,

wherein the optimization procedure comprises rearranging policy statements to improve the performance,

wherein for a field-operator pair in an associated client DLL, the rearranging comprises:

determining when a statement is processing intensive; and

rearranging statements of the policy to check at least some of the processing intensive statements after less processing intensive statements.

17. The method of claim 16 wherein the optimization procedure further comprises eliminating at least some repeated statements including repeated portions of the statements, and

wherein as part of the compiling, the method comprises generating an optimized policy transmission language with the eliminated statements and eliminated repeated portions of the statements.

18. The method of claim 16 further comprising:

marking route attributes in a batch of routes as invariant for ones of the route attributes that share values across differing route prefixes; and

caching results for statements associated with the marked attributes when executing the compiled policy transmission language,

wherein upon continued execution of the compiled policy transmission language, using the cached results for evaluations of subsequent statements in the compiled policy transmission language which reference the marked attributes.

19. A policy compiler comprising:

means for generating a configuration data abstraction layer of a routing policy, the configuration data abstraction layer to map configuration to an intermediate layer comprising fields, operators and arguments; and

means for verifying the intermediate layer against a set of verification rules for one or more client protocols including versions thereof, the policy repository to generate compiled policy transmission language for use by the one or more client protocols including versions thereof,

wherein the means for verifying is to verify the intermediate layer against the set of verification rules for a particular one or more of a plurality of client

protocols including versions of the one or more of the plurality of client protocols, and

wherein the means for verifying is to generate the compiled policy transmission language for use by the one or more client protocols including versions thereof.

20. A machine-readable medium that provides instructions, which when executed by one or more processors, cause the processors to perform operations comprising generating compiled policy for execution by one or more client protocols including one or more versions of the client protocols,

wherein the instructions, when further executed by one or more of the processors cause the processors to perform operations further comprising:

generating a configuration data abstraction layer for a routing policy to map configuration to an intermediate layer;

verifying the intermediate layer against a set of rules for the one or more client protocols including the one or more versions thereof; and

generating compiled policy transmission language when statements of the intermediate layer are verified against the set of rules for the one or more client protocols including the one or more versions thereof.

21. A method for optimizing a compiled policy, the method comprising:

marking statements in a compiled policy as invariant for ones of the route attributes that share values across differing route prefixes; and

caching results for statements associated with the marked attributes when executing the compiled policy,

wherein upon continued execution of the compiled policy, using the cached results for evaluations of subsequent statements in the policy which reference the marked attributes.

22. The method of claim 21 wherein marking comprises statements in a compiled representation of the policy as invariant when one or more attributes do not change with respect to received routes that are to be executed with respect to an associated policy.

23. The method of claim 22 wherein caching comprises temporarily storing results of executing statements associated with the marked attributes when executing the compiled policy, and

wherein using comprises using the cached results for the marked attributes to obtain a result for a route, the result being one of a modified route, an accepted and unchanged route, or dropped route.

24. The method of claim 23 further comprising:

prior to the marking, receiving a route update message from a peer, the route update message being associated one of a plurality of client protocols and associated with a version of one of the client protocols,

wherein the route update message comprises a plurality of updated routes wherein at least some of the updated routes with differing prefixes have attributes of identical values, and

wherein marking comprising marking the attributes of identical values of the updated routes.

25. The method of claim 24 wherein the executing comprises running the compiled policy on an execution engine associated with a client protocol to apply the updated routes,

wherein execution of the compiled policy, including the use of the cached results, results in reduced processing operations for the associated execution engine.

26. The method of claim 25 wherein the caching comprising caching results of comparison statements that reference route attributes marked as invariant, and

wherein the executing comprises refraining from executing the comparison statements in the compiled policy having cached results.

27. The method of claim 21 wherein the compiled policy is in a policy transmission language, and

wherein the method further comprises, prior to the marking, generating the compiled policy for execution by one or more client protocols or more than one version the client protocols, the generating comprising:

generating a configuration data abstraction layer for a routing policy to map configuration to an intermediate layer;

verifying the intermediate layer against a set of rules for the one or more client protocols including the one or more versions thereof; and

generating compiled policy transmission language when statements of the intermediate layer are verified against the set of rules for the one or more client protocols including the one or more versions thereof.

28. A compiler comprising:

one or more processors to implement a plurality of client protocols including versions thereof to mark statements in a route update message as invariant for route attributes that share values across differing route prefixes; and

an execution engine associated with one of the client protocols including versions thereof to cache results for statements associated with the marked attributes when executing the compiled policy,

wherein upon continued execution of the compiled policy, the execution engine is to use the cached results for evaluations of subsequent statements in the policy which reference the marked attributes.

29. The compiler of claim 28 wherein one of the client protocols including versions thereof is to mark route attributes in a complied batch of routes as invariant when one or more attributes do not change with respect to received routes that are to be executed with respect to an associated routing policy.

30. The compiler of claim 29 wherein as part of caching, the execution engine is to temporarily store execution results for statements associated with the marked attributes when executing the compiled policy, and

wherein the execution engine is to use the cached results for the marked attributes to obtain a result for a route, the result being one of a modified route, an accepted and unchanged route, or dropped route.

31. The compiler of claim 30 wherein one of the client protocols including versions thereof is to receive a route update message from a peer router, the route update message being associated one of the client protocols and associated with one of the versions of the client protocols,

wherein the route update message comprises a plurality of updated routes wherein at least some of the updated routes with differing prefixes have attributes of identical values, and

wherein the one of the client protocols is to further mark the attributes of identical values.

32. The compiler of claim 31 wherein the execution engine is to run the compiled policy using the received routes,

wherein when executing the compiled policy, the execution engine's use of the cached results is to reduce processing operations for the execution engine.

33. The compiler of claim 32 wherein the execution engine is to cache results of comparison statements that reference route attributes marked as invariant, and is to refrain from executing the comparison statements in the compiled policy having the cached results.

34. The compiler of claim 28 further comprising a policy repository and a system controller, and

wherein the system controller is to generate a configuration data abstraction layer of a routing policy, the configuration data abstraction layer to map configuration to an intermediate layer comprising fields, operators and arguments,

wherein the policy repository is to verify the intermediate layer against a set of verification rules for one or more of the client protocols including versions thereof, and

wherein the policy repository is to generate the compiled policy in the policy transmission language for use by the one or more client protocols including versions thereof.

35. A compiler comprising:

a plurality of means for marking route attributes in a route and statements in a complied policy as invariant for ones of the route attributes that share values across differing route prefixes;

means, associated with the means for marking, to cache results for statements associated with the marked attributes; and

means for executing the compiled policy with the marked route attributes,

wherein upon continued execution of the compiled policy, the means for executing is to use the cached results for evaluations of subsequent statements in the policy which reference the marked attributes.

36. A machine-readable medium that provides instructions, which when executed by one or more processors, cause the processors to perform operations comprising:

marking route attributes in a compiled policy as invariant for ones of the route attributes that share values across differing route prefixes; and

caching results for statements associated with the marked attributes when executing the compiled policy,

wherein upon continued execution of the compiled policy, using the cached results for evaluations of subsequent statements in the policy which reference the marked attributes,

wherein the marking comprising marking route attributes in a compiled representation of the policy as invariant when one or more attributes do not change with respect to received routes that are to be executed with respect to an associated policy, and

wherein the caching comprises temporarily storing results of the executing for statements associated with the marked attributes when executing the compiled policy, and

wherein using comprises using the cached results for the marked attributes to obtain a result for a route, the result being one of a modified route, an accepted and unchanged route, or dropped route.

37. A method for running at least first and second versions of a client protocol on a router, the method comprising:

associating a dynamic registration process with each of the at least first and second versions of the client protocol,

wherein the dynamic registration processes are to provide a policy repository with a location of a dynamic link library (DLL) for an associated one of the versions of the client protocol,

wherein the dynamic registration processes are to further provide the policy repository with a registration location for configuration space to be used by a routing policy for use when registering with the associated one of the versions of the client protocols, and

wherein each of the versions of the client protocol has a configuration space and a DLL associated therewith.

38. The method of claim 37 wherein when a routing policy is to be registered with the second version of the client protocol while the first version of the client protocol is running the router, the method further comprises:

verifying policy statements of the routing policy with verification rules in the DLL associated with the second version of the client protocol; and

using the configuration space associated with the second version of the client protocol for the verification.

39. The method of claim 38 wherein first and second DLLs are associated respectively with the first and second versions of the client protocol,

wherein the DLLs comprise verification rules for the associated version of the client protocol, and

wherein the router is to support the addition of the second version of the client protocol while the first version is running on the router.

40. The method of claim 39 further comprising associating an additional dynamic registration process for each of at least one additional client protocol.

41. The method of claim 40 wherein the at least first and second versions of the client protocol include at least two versions of a border gateway protocol (BGP), and

wherein the at least one additional client protocol comprises at least one of an open-shortest-path-first (OSPF) client protocol and an intermediate system to intermediate system (IS-IS) client protocol.

42. The method of claim 41 wherein when the policy statements of the routing policy are verified, the method comprises;

moving at least some peer routers to the second version of the client protocol; and

removing the at least some peer routers from the first version of the client protocol.

43. The method of claim 42 wherein the dynamic registration processes comprise software agents operating within the router.

44. The method of claim 35 further comprising:

querying, by a parser, the policy repository for capabilities of configuration associated with the second version of the client protocol when the second version of the client protocol is being added.

45. A router comprising:

a policy repository; and

one or more processors to implement dynamic registration processes, each of the processes associated with at least first and second versions of a client protocol,

wherein the dynamic registration processes are to provide the policy repository with a location of a dynamic link library (DLL) for an associated one of the versions of the client protocol,

wherein the dynamic registration processes are to further provide the policy repository with a registration location for configuration space to be used by a routing policy for use when registering with the associated one of the versions of the client protocols, and

wherein each of the versions of the client protocol has a configuration space and a DLL associated therewith.

46. The router of claim 45 wherein when a routing policy is to be registered with the second version of the client protocol while the first version of the client protocol is running the router, the second version of the client protocol is to verify policy statements of the routing policy with verification rules in the DLL associated with the second version of the client protocol and use the configuration space associated with the second version of the client protocol for the verification.

47. The router of claim 46 wherein first and second DLLs are associated respectively with the first and second versions of the client protocol,

wherein the DLLs comprise verification rules for the associated version of the client protocol, and

wherein the router is to support the addition of the second version of the client protocol while the first version is running on the router.

48. The router of claim 47 wherein the one or more processors are to further implement additional dynamic registration processes associated for each of at least one additional client protocol.

49. The router of claim 48 wherein the at least first and second versions of the client protocol include at least two versions of a border gateway protocol (BGP), and

wherein the at least one additional client protocol comprises at least one of an open-shortest-path-first (OSPF) client protocol and an intermediate system to intermediate system (IS-IS) client protocol.

50. The router of claim 49 wherein when the policy statements of the routing policy are verified, the second version of the client protocol is to be requested to move at least some peer routers to the second version of the client protocol and is to be requested to remove the at least some peer routers from the first version of the client protocol.

51. The router of claim 50 wherein the dynamic registration processes comprise software agents operating on the one or more processors.

52. The router of claim 51 further comprising a parser to query the policy repository for capabilities of configuration associated with the second version of the client protocol when the second version of the client protocol is being added.

53. A router comprising:

means for coordinating routing policies; and

means for associating dynamic registration processes with at least first and second versions of a client protocol,

wherein the dynamic registration processes are to provide means for coordinating with a location of a dynamic link library (DLL) for an associated one of the versions of the client protocol,

wherein the dynamic registration processes are to further provide the means for coordinating with a registration location for configuration space to be

used by a routing policy for use when registering with the associated one of the versions of the client protocols, and

wherein each of the versions of the client protocol has a configuration space and a DLL associated therewith,

wherein when a routing policy is to be registered with the second version of the client protocol while the first version of the client protocol is running the router, the second version of the client protocol includes means for verifying policy statements of the routing policy with verification rules in the DLL associated with the second version of the client protocol, and means for using the configuration space associated with the second version of the client protocol for the verification.

54. A machine-readable medium that provides instructions, which when executed by one or more processors, cause the processors to perform operations for running at least first and second versions of a client protocol on a router, the operations comprising:

associating a dynamic registration process with each of the at least first and second versions of the client protocol,

wherein the dynamic registration processes are to provide a policy repository with a location of a dynamic link library (DLL) for an associated one of the versions of the client protocol,

wherein the dynamic registration processes are to further provide the policy repository with a registration location for configuration space to be used by a routing policy for use when registering with the associated one of the versions of the client protocols, and

wherein each of the versions of the client protocol has a configuration space and a DLL associated therewith.

55. A method of verifying statements of a routing policy prior to compiling the routing policy, the method comprising:

generating libraries for attach points associated with one or more versions of one or more client protocols, the libraries to include capabilities for the one or more versions of the one or more client protocols; and

individually checking statements of a routing policy against the capabilities of one or more of the attach points.

56. The method of claim 55 wherein the libraries describe fields supported by an associated one of the versions of the one or more client protocols,

wherein the libraries further describe operators supported by the fields for the associated one of the versions of the one or more client protocols, and

wherein the libraries include a verification function for field-operator combinations associated with one of the versions of the one or more client protocols.

57. The method of claim 56 further comprising individually checking the statements when the routing policy is initially being defined.

58. The method of claim 57 further comprising informing a policy repository to load one of the libraries associated with a particular one of the attach points associated with one of the versions of the one or more client protocols,

wherein the informing is performed by a dynamic registration process associated with one of the versions of the one or more client protocols.

59. The method of claim 58 wherein the versions of the one or more client protocols have one or more attach points, the attach points comprising at least some of incoming neighbor points, outgoing neighbor points, aggregation points and dampening points.

60. The method of claim 58 further comprising:

during the checking, providing an indication when a statement of the routing policy currently being checked is not supported by the capabilities of one of the attach points; and

receiving a selection by a user, the selection to direct a policy repository to either skip the statement not supported by an attach point, or reject the policy when the statement is not supported by an attach point.

61. The method of claim 60 wherein the indication is selectable and includes one of ignoring the statement when an operation is not supported by an attach point, ignoring and warning that the operation is not supported by an attach point, and rejecting the policy when the operation is not supported by an attach point.

62. The method of claim 60 further comprising:

compiling the routing policy when either all statements are verified or when statements not supported by an attach point are skipped; and

when executing the compiled routing policy by an execution engine of a version of a client protocol, skipping the statements not supported by an associated attach point,

wherein the skipped statements comprise comparison operators.

63. The method of claim 61 further comprising performing a verification for statements of the policy having comparison operators when applying a compiled policy to an attach point.

64. The method of claim 55 further comprising providing a notification when all statements of a policy are verified for at least one attach point, the notification to include which of the one or more attach points the policy is valid.

65. A policy compiler comprising:

a policy repository to store capabilities for attach points,

one or more processors to implement a plurality of dynamic registration processes to inform the policy repository of the capabilities for the attach points, the dynamic registration processes being associated with a version of a client protocol; and

libraries for attach points associated with one or more versions of one or more client protocols, the libraries to include capabilities for the one or more versions of the one or more client protocols,

wherein the policy repository is to check statements of a routing policy against the capabilities of one or more of the attach points during generation of the policy.

66. The policy compiler of claim 65 wherein the libraries describe fields supported by an associated one of the versions of the one or more client protocols,

wherein the libraries further describe operators supported by the fields for the associated one of the versions of the one or more client protocols, and

wherein the libraries include a verification function for field-operator combinations associated with one of the versions of the one or more client protocols.

67. The policy compiler of claim 66 wherein the policy repository is to further individually check statements when the routing policy is initially being defined.

68. The policy compiler of claim 67 wherein one of the dynamic registration processes associated with one of the versions of the one or more client protocols is to inform a policy repository to load one of the libraries associated with a particular one of the attach points associated with one of the versions of the one or more client protocols.

69. The policy compiler of claim 68 wherein the versions of the one or more client protocols have one or more attach points, the attach points comprising at least some of incoming neighbor points, outgoing neighbor points, aggregation points and dampening points.

70. The policy compiler of claim 68 further comprising an I/O, wherein during the checking, the policy compiler is to provide an indication when a statement of the routing policy currently being checked is not supported by the capabilities of an attach point, and

wherein a selection by a user is to be received through the I/O, the selection to direct the policy repository to either skip the statement not supported by an attach point, or reject the policy when the statement is not supported by an attach point.

71. The policy compiler of claim 70 wherein the indication is selectable and includes one of ignoring the statement when an operation is not supported by an attach point, ignoring warning that the operation is not supported by an attach point, and rejecting the policy when the operation is not supported by an attach point.

72. The policy compiler of claim 70 wherein the policy repository is to compile the routing policy when either all statements are verified or when statements not supported by an attach point are skipped, and

wherein the policy compiler further comprises an execution engine associated with a version of a client protocol, the execution engine to execute the compiled routing policy and to skip statements not supported by an associated attach point.

73. A policy compiler comprising:

means for storing capabilities for attach points,

means for informing a policy repository of the capabilities for the attach points, the means for informing being associated with a version of a client protocol;

means for generating libraries for attach points associated with one or more versions of one or more client protocols, the libraries to include capabilities for the one or more versions of the one or more client protocols;

means for checking statements of a routing policy against the capabilities of one or more of the attach points during generation of the policy;

means for providing an indication when a statement of a routing policy currently being checked is not supported by the capabilities of one of the attach points; and

means for receiving a selection by a user, the selection to direct the means for checking to either skip the statement not supported by the attach point, or reject the policy when the statement is not supported by the attach point.

74. A machine-readable medium that provides instructions, which when executed by one or more processors, cause the processors to perform operations for verifying statements of a routing policy, the operations comprising:

generating libraries for attach points associated with one or more versions of one or more client protocols, the libraries to include capabilities for the one or more versions of the one or more client protocols; and

individually checking statements of a routing policy against the capabilities of one or more of the attach points.

75. A method for transitioning between routing policies, wherein a first configuration state is associated with the first route policy, wherein a second configuration state is associated with a second route policy, and wherein the first configuration state is a current configuration state of a router, the method comprising:

grouping configuration elements of the second route policy into policy statements and sets;

verifying the grouped configuration elements against capabilities with verification rules associated with one or more versions of one or more client protocols;

compiling statements of the second route policy when verified for at least one of the one or more versions of the one or more client protocols; and

notifying the at least one of the one or more versions of the one or more client protocols that the second route policy is to take effect.

76. The method of claim 75 further comprising, after the compiling, overwriting a compiled version of the first route policy with the compiled second route policy, the second route policy being in a compiled policy transmission language.

77. The method of claim 76 wherein in response to the notifying, the at least one of the one or more versions of the one or more client protocols is to execute the compiled second route policy, and

wherein the router is to apply the second route policy to attach points associated with the at least one of the one or more versions of the one or more client protocols.

78. The method of claim 77 wherein prior to and during the grouping, the verifying, the compiling and the notifying, the method further comprising running the first route policy.

79. The method of claim 75 further comprising receiving statements representing the second route policy from an operator.

80. The method of claim 77 wherein the second route policy comprises policy statements that identify fields, operators and arguments, and

wherein verifying comprises, for each policy statement:

verifying fields for the one or more versions of the one or more client protocols;

verifying field-operator pairings for the one or more versions of the one or more client protocols; and

verifying arguments for each field-operator pairing for the one or more versions of the one or more client protocols.

81. A policy compiler to transition a router from a first configuration state associated with the first route policy to a second configuration state associated with a second route policy, the policy compiler comprising:

a policy repository to group configuration elements of the second route policy into policy statements and sets, and to verify the grouped configuration elements against capabilities with verification rules associated with one or more versions of one or more client protocols, the policy repository to further compile statements of the second route policy when verified for at least one of the one or more versions of the one or more client protocols; and

a system controller to notify the at least one of the one or more versions of the one or more client protocols that the second route policy is to take effect.

82. The policy compiler of claim 81 wherein after compiling, the policy repository is to overwrite a compiled version of the first route policy with the compiled second route policy, the second route policy being in a compiled policy transmission language.

83. The policy compiler of claim 82 wherein the at least one of the one or more versions of the one or more client protocols is to execute the compiled second route policy, and

wherein the router is to apply the second route policy to attach points associated with the at least one of the one or more versions of the one or more client protocols.

84. The policy compiler of claim 83, wherein prior to and during the policy repository grouping, verifying, and compiling, the router is to run the first route policy.

85. The policy compiler of claim 84 further comprising an I/O to receive the statements representing the second policy from an operator.

86. The policy compiler of claim 85 wherein the second route policy comprises policy statements that identify fields, operators and arguments, and

wherein the policy repository is to verify, for route policy statement, fields for the one or more versions of the one or more client protocols, field-operator pairing for the one or more versions of one or more client protocols, and arguments for each field-operator pairing for the one or more versions of the one or more client protocols.

87. A policy compiler to transition a router from a first configuration state associated with the first route policy to a second configuration state associated with a second route policy, the policy compiler comprising:

means for grouping configuration elements of the second route policy into policy statements and sets;

means for verifying the grouped configuration elements against capabilities with verification rules associated with one or more versions of one or more client protocols;

means for compiling statements of the second route policy when verified for at least one of the one or more versions of the one or more client protocols; and

means for notifying the at least one of the one or more versions of the one or more client protocols that the second route policy is to take effect.

88. A machine-readable medium that provides instructions, which when executed by one or more processors, cause the processors to perform operations for transitioning between routing policies, wherein a first configuration state is associated with the first route policy, wherein a second configuration state is associated with a second route policy, and wherein the first configuration state is a current configuration state of a router, the operations comprising:

grouping configuration elements of the second route policy into policy statements and sets;

verifying the grouped configuration elements against capabilities with verification rules associated with one or more versions of one or more client protocols;

compiling statements of the second route policy when verified for at least one of the one or more versions of the one or more client protocols; and

notifying the at least one of the one or more versions of the one or more client protocols that the second route policy is to take effect.